Course – 5 Date: 01/08/2024

WEEK 1: Recurrent Neural Networks (RNNs)

Example:

x: Word1 Word2 Word3 word4 ... word9

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<t>}$ ... $x^{<9>}$

$T_x = 9$ (Length of input sequence)

y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ ... $y^{<9>}$

$T_y = 9$ (Length of output sequence)

T_x and T_y need not be equal always.

$x^{(i)<t>}$ $T_x^{(i)}$

i^{th} example length of sequence of i^{th} example

$y^{(i)<t>}$ $T_y^{(i)}$

Date:

Representing words :

x : Sentence of words

Vocabulary

a	1
aaron	2
⋮	⋮
and	367
⋮	⋮
harry	4075
⋮	⋮
potter	6830
⋮	⋮
zulu	10000

Usually size is 30k, 50k and even 100k is not uncommon.

Size: 10,000

Then using one_hot( ) all words can be assigned a vector where 1 is assigned to their respective index

harry $x^{<1>}$ → 4075

potter $x^{<2>}$ → 6830

$x^{<t>}$ (x, y)

x → y

one_hot vectors

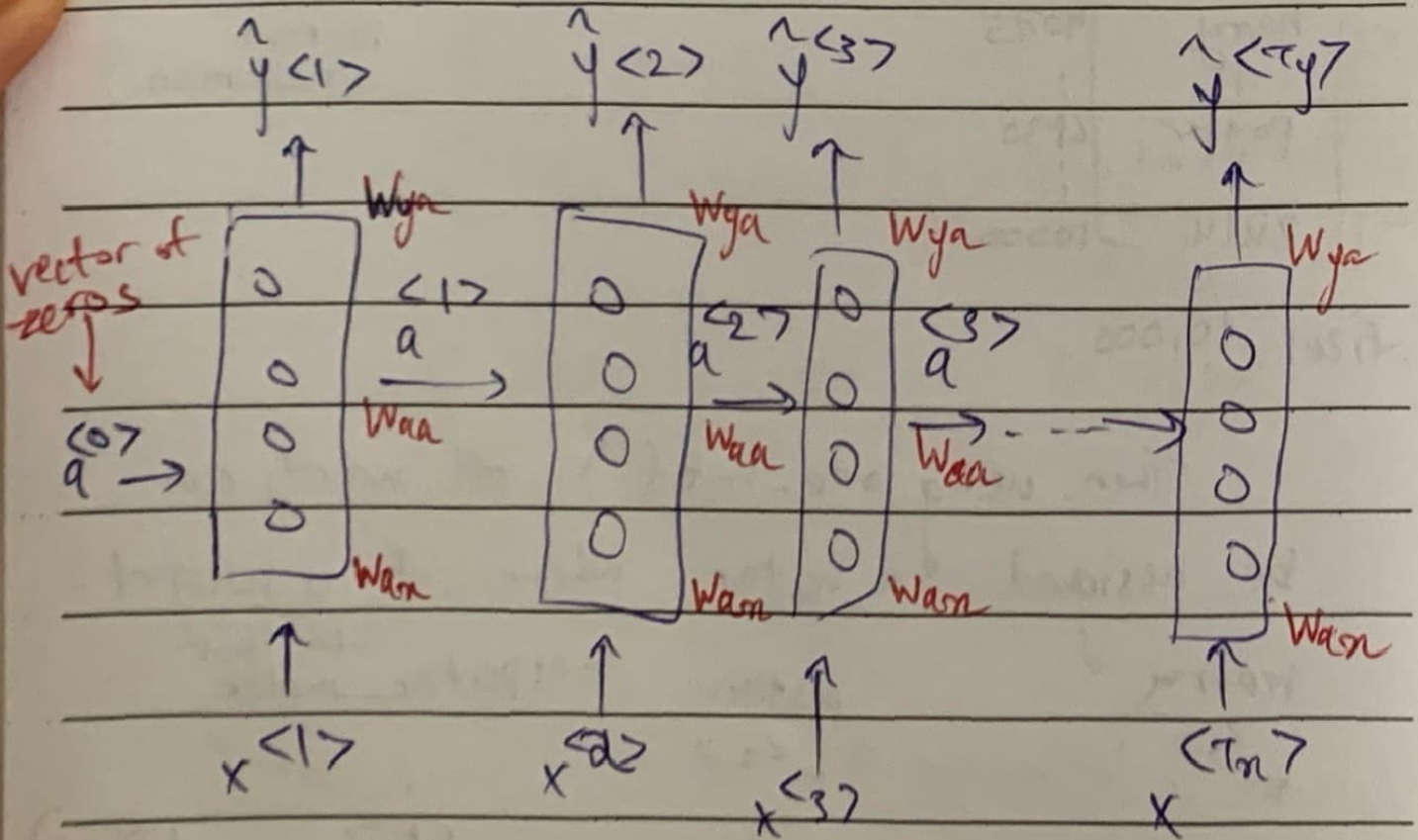
Date:

Why not a standard NN?

Problems:

① T_x and T_y will be different in every example, so doesn't work very well.

② Doesn't share features learned across different positions of text.



Here $T_x = T_y$

Architecture will change a bit if $T_x \neq T_y$

Date:

→ The limitation of this RNN is that to predict the output at the current time step, it uses info from the previous time steps but cannot use the later time steps.

$a^{\langle 0 \rangle} = \vec{0}$

$a^{\langle 1 \rangle} = g_1(W_{aa} a^{\langle 0 \rangle} + W_{ax} x^{\langle 1 \rangle} + b_a)$

This indicates that it is used for computation of "a" and will be multiplied by "x"

Most commonly used is tanh() and rarely ReLU

$\hat{y}^{\langle 1 \rangle} = g_2(W_{ya} a^{\langle 1 \rangle} + b_y)$

same here

for binary classificat^n sigmoid otherwise Softmax but depends on requirement

$$a^{\langle t \rangle} = g(W_{aa} a^{\langle t-1 \rangle} + W_{ax} x^{\langle t \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = g(W_{ya} a^{\langle t \rangle} + b_y)$$

Date:

Simplified RNN notation:

$$a^{<t>} = g\left(W_a \left[a^{<t-1>}, x^{<t>}\right] + b_a\right)$$

$$\hat{y}^{<t>} = g\left(W_y a^{<t>} + b_y\right)$$

If $W_{aa} : 100$

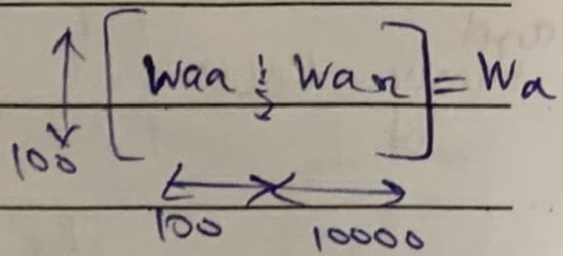
$W_{ax} : 10000$

then

$W_{aa} : 100, 100$

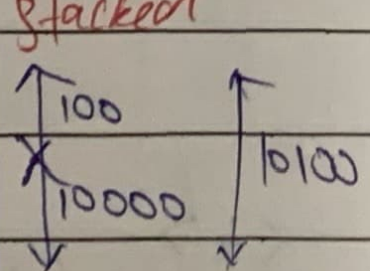
$W_{ax} : 100, 10000$

Stacked together as



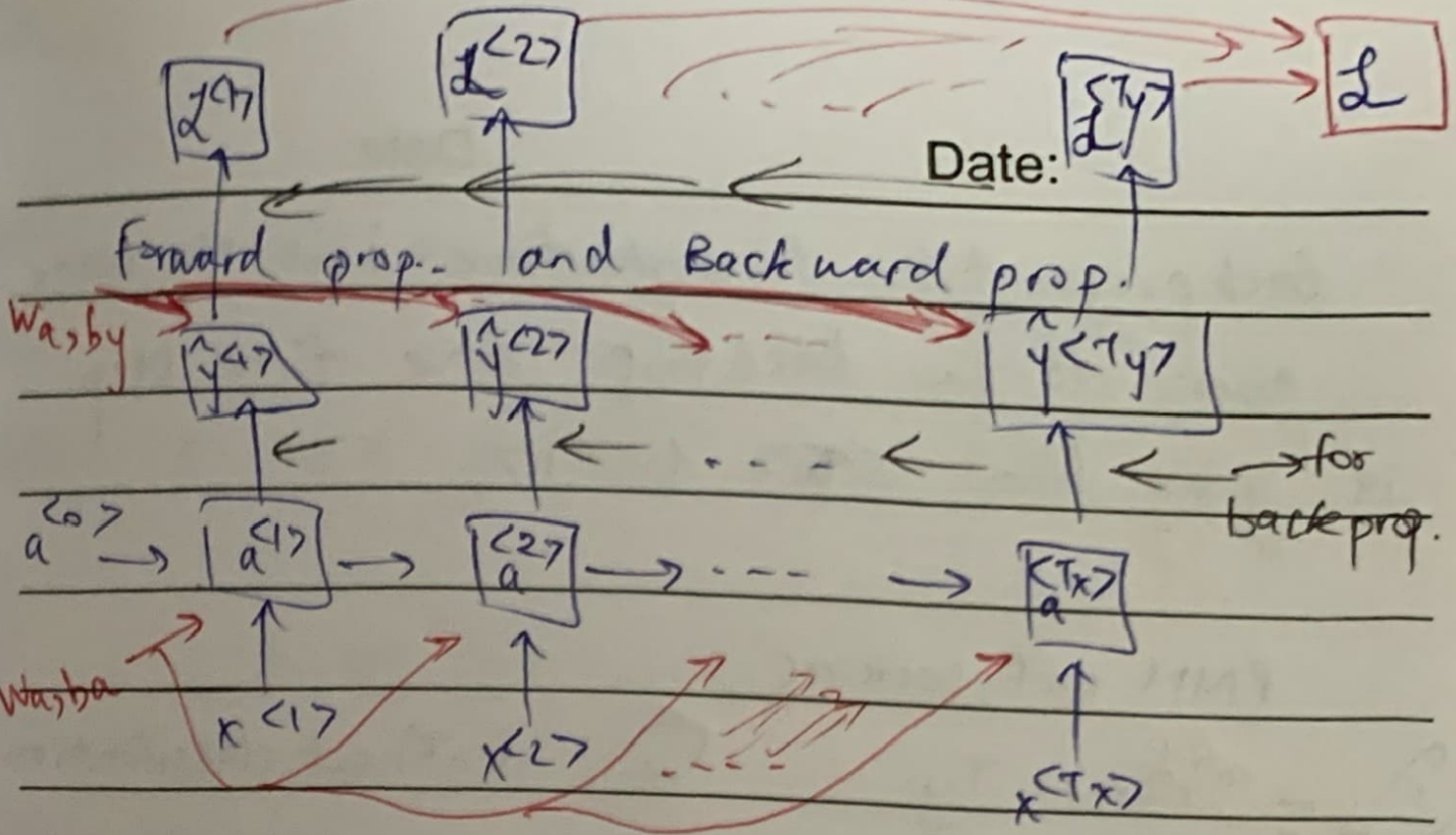
$\therefore W_a : (100, 10100)$

$$\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} = \begin{bmatrix} a^{<t-1>} \\ \cdots \\ x^{<t>} \end{bmatrix}$$



$$\therefore \begin{bmatrix} W_{aa} ; W_{ax} \end{bmatrix} \begin{bmatrix} a^{<t-1>} \\ \cdots \\ x^{<t>} \end{bmatrix} = W_{aa} a^{<t-1>} + W_{ax} x^{<t>}$$

Date:



$$\mathcal{L}^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}) = -y^{\langle t \rangle} \log \hat{y}^{\langle t \rangle} - (1 - y^{\langle t \rangle}) \log (1 - \hat{y}^{\langle t \rangle})$$

loss for single time step single word

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$

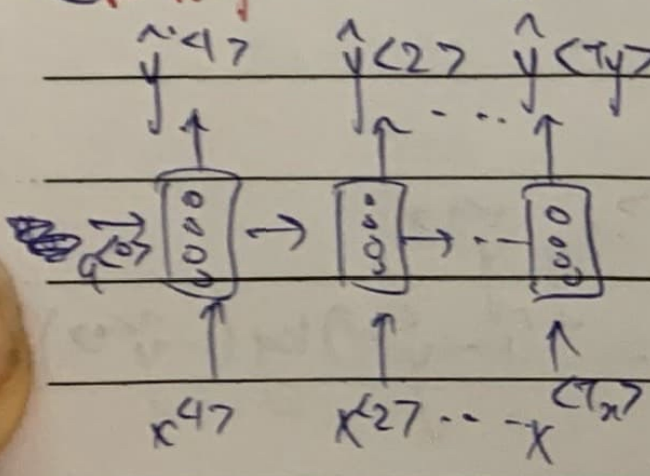
loss for entire sequence

Date:

Backpropagation through time ; is the name given to the backprop since time step is going from <Tx> to <1>

RNN architectures :

(1) Name entity recognition $T_x = T_y$



Many-to-Many

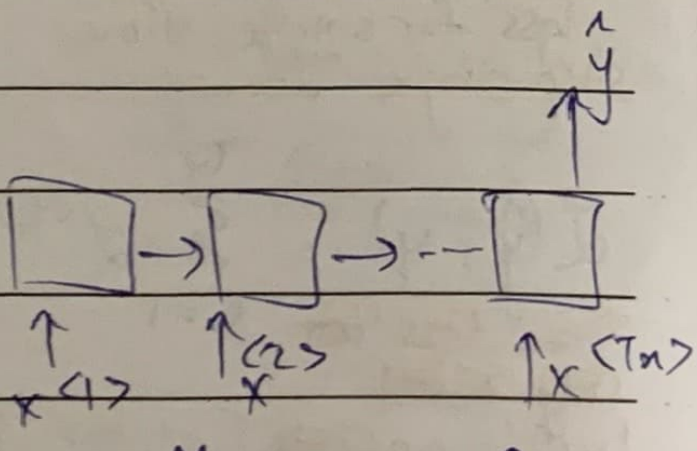
Many inputs

Many outputs

(2) Sentiment classification

X = text sequence

y = 0/1 or 1-5

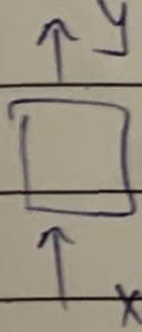


Many-to-One

Many input

Outputs one number

(3)



One-to-One (Similar almost same as standard NN)

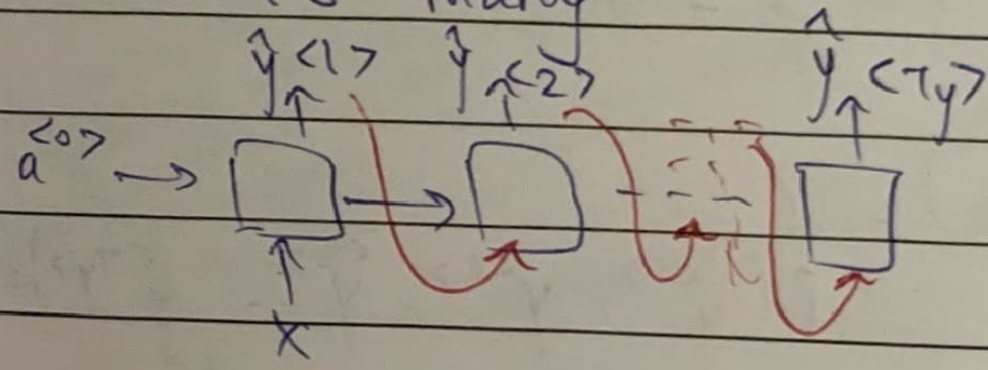
Date:

(4) Music generation

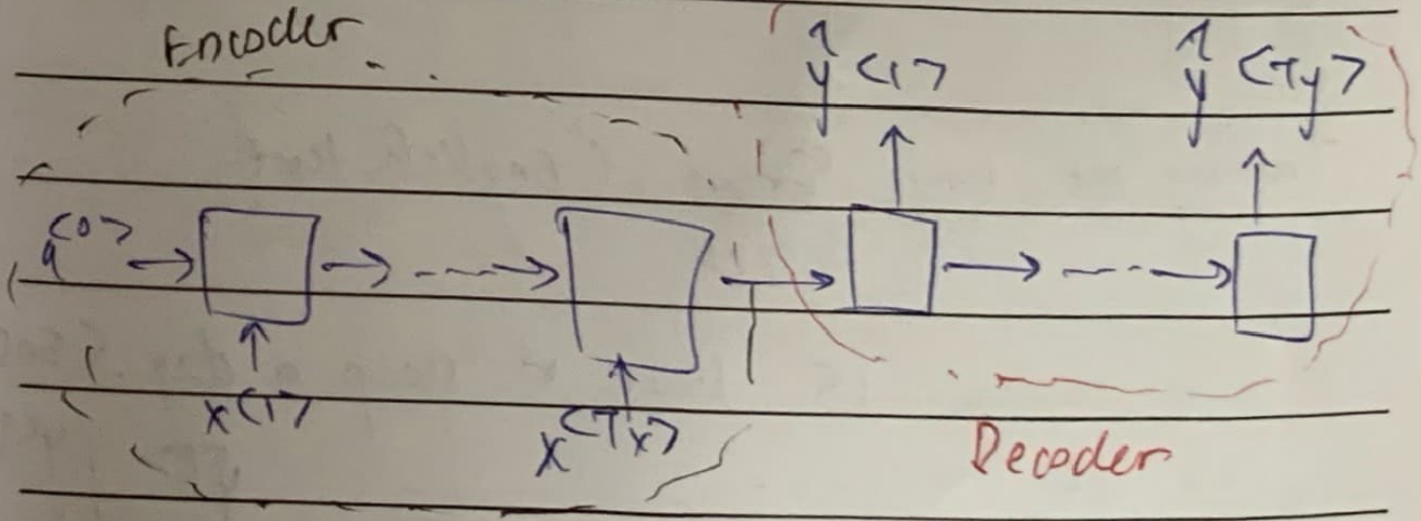
$X \longrightarrow y^{\langle 1 \rangle} y^{\langle 2 \rangle} \cdots\cdots y^{\langle T_y \rangle}$

X can even be $X = \phi$

One-to-many



(5) Machine translation



Many-to-many $T_x \neq T_y$

Date:

What is language modelling?

→ Speech recognition:

The apple and pair salad.

The apple and pear salad

$P(1^{st}) = 3.2 \times 10^{-13}$

$P(2^{nd}) = 5.7 \times 10^{-10}$

$P(\text{sentence}) = ?$

$P(y^{<1>}, y^{<2>}, y^{<3>} \ldots y^{<Ty>}) =$

Language modelling with a RNN:

Training set: large ~~corpus~~ corpus of English text.

Cats average 15 hours of sleep a day. <EOS>

$y^{<1>}$ $y^{<2>}$ .... $y^{<8>}$ $y^{<9>}$

End of Sentence token

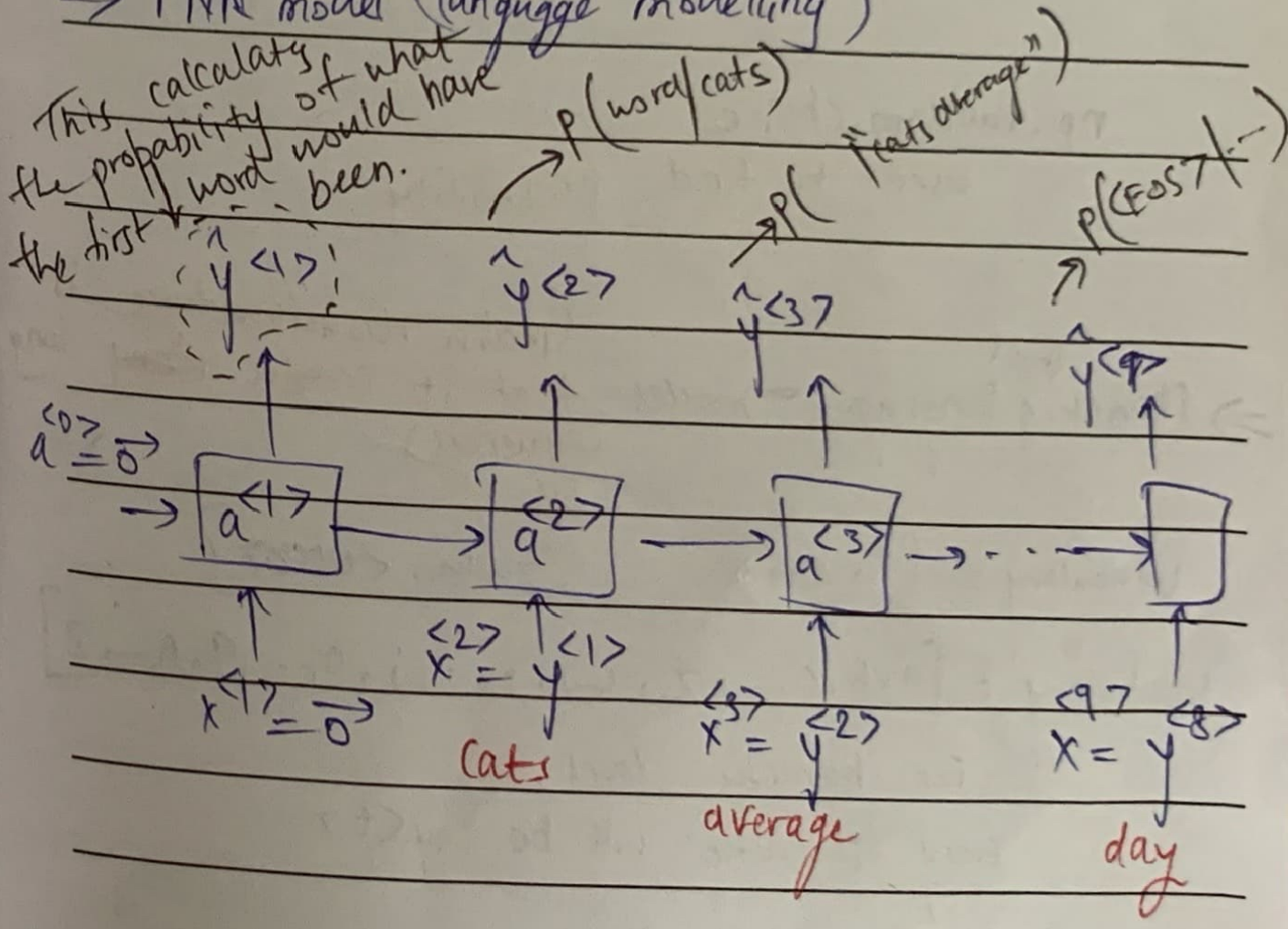
$P(y^{<1>}, y^{<2>}, y^{<3>}) = P(y^{<1>}) \cdot P(y^{<2>} | y^{<1>}) \cdot P(y^{<3>} | y^{<2>}, y^{<1>})$

Date:

The Egyptian Mau is a breed of cat. <EOS>

<UNK>
Unknown word
(not in the dictionary-vocabulary of 10k words used)

⇒ RNN model (langugge modelling)

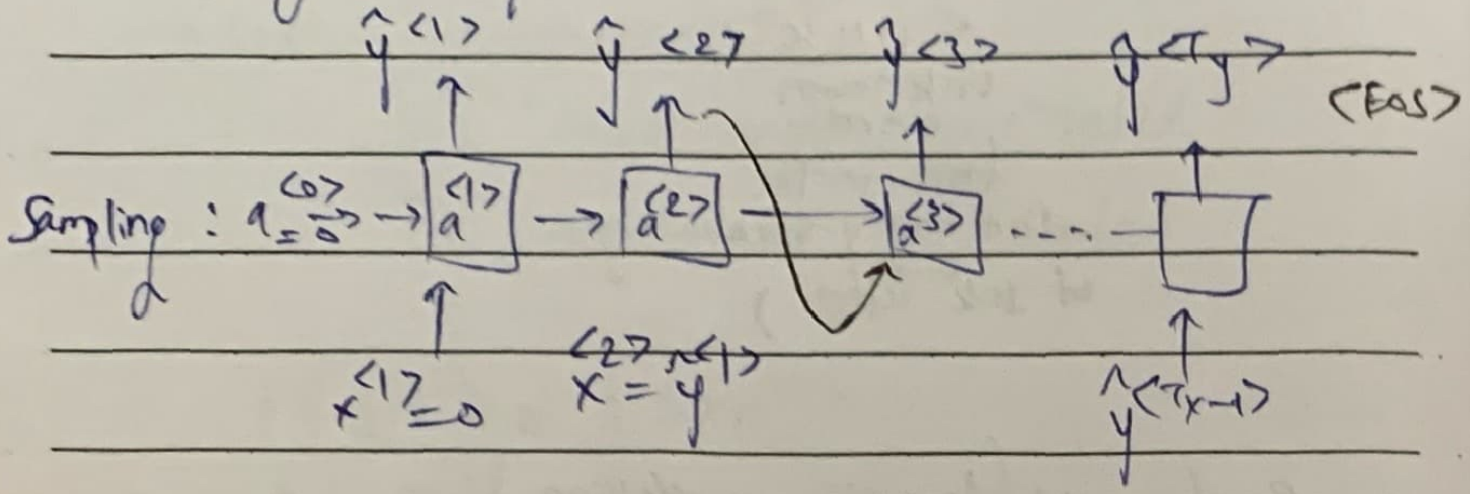


$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$

$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$

Date:

Sampling a sequence from a trained RNN;



np.random.choice

used to find probability

⇒ Character level language model: (main disadvantage is that it creates ~~many~~ long sequences)

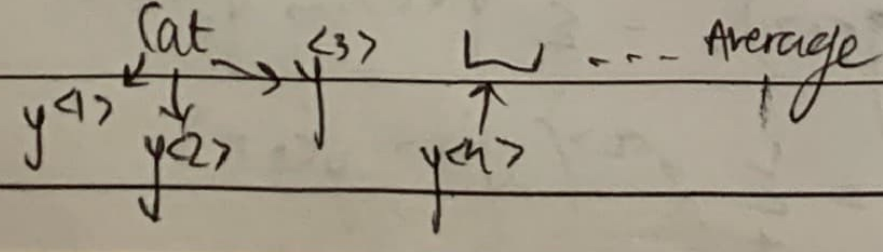
Vocabulary = [a, aaron, ..., zulu, <UNK>]

Vocabulary = [a, b, c ..., z, ␣, ., ,, ;, 0, ..., 9, A, ..., Z]

for character level,

every character will be $y^{<t>}$

like:



Date:

Character level not good at long range dependencies. and more computationally expensive to train.

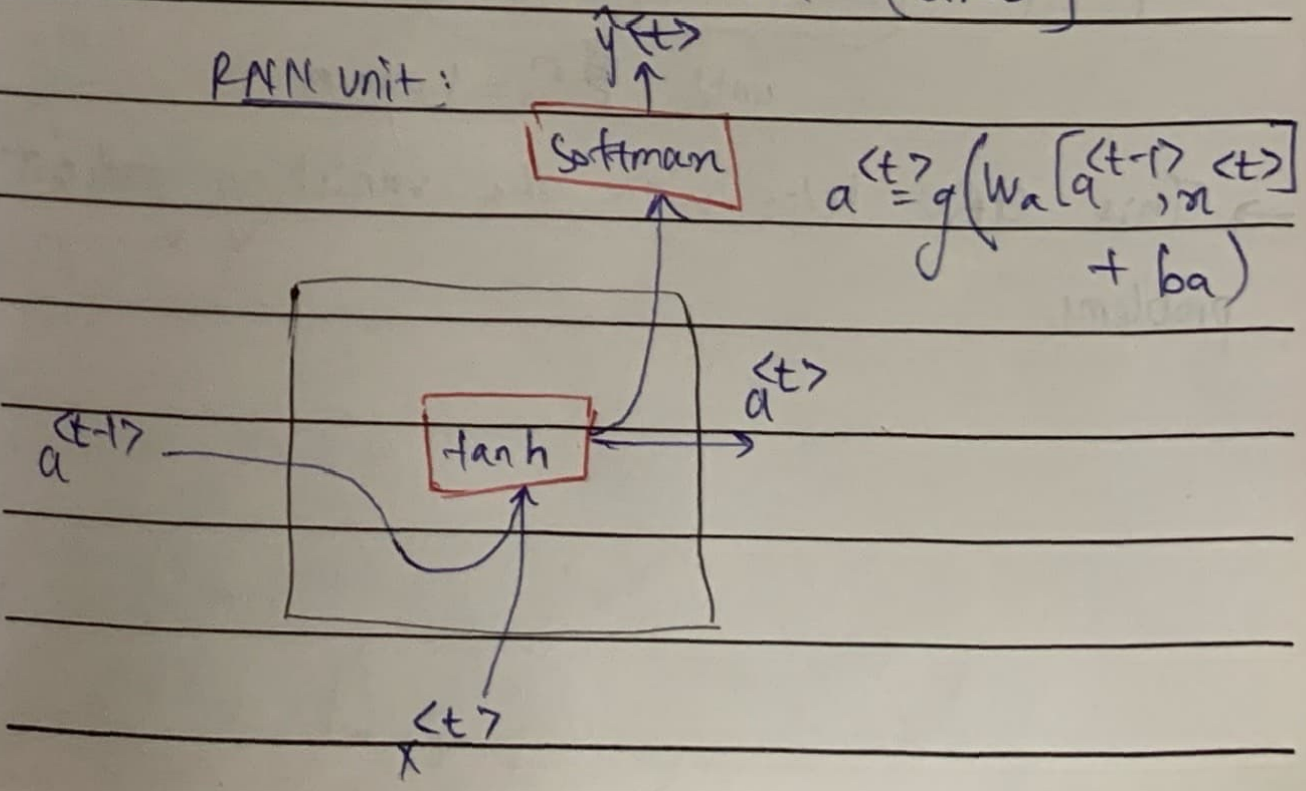
=> Vanishing gradients with basic RNNs:

The RNNs are not very good at capturing long term dependencies.

To manage Exploding gradients, use (gradient clipping)

=> Gated Recurrent Unit [GRU]

RNN unit:



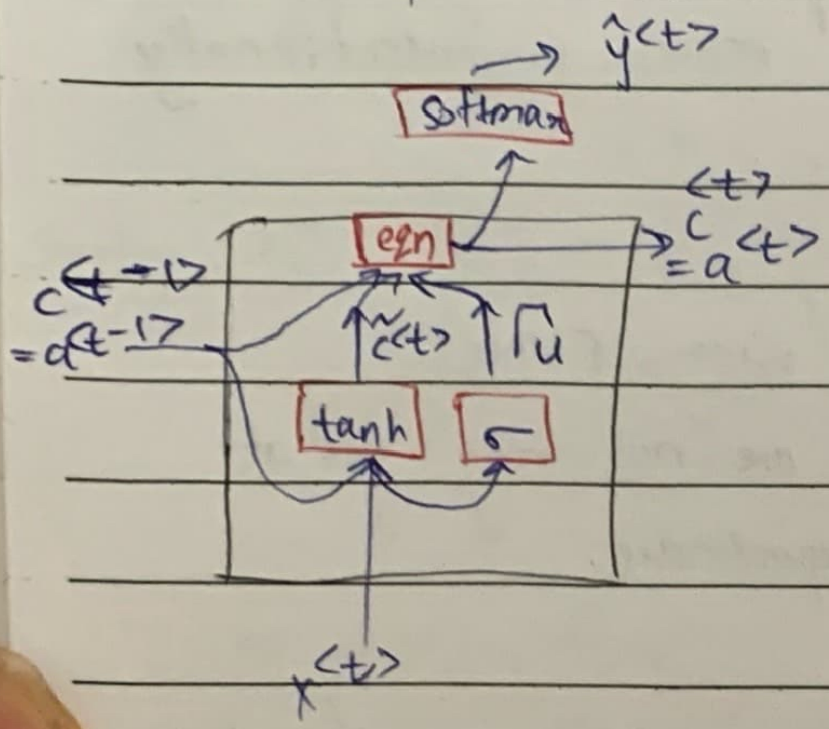
Γ_u = gamma u

↓ "update"

In eqn, * is element-wise mulp.

Date:

GRU (simplified):



$c^{<t>}$ = memory cell

$c^{<t>} = a^{<t>}$

$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$

$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$

↳ always almost 0/1

eqn → $c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1-\Gamma_u) * c^{<t-1>}$

∴ if $\Gamma_u = 1$ $\Gamma_u = 0$ $\Gamma_u = 0$ $\Gamma_u = 0$ ... $\Gamma_u = 1$

$c^{<t>} = 1$ (doesn't update until $\Gamma_u = 1$ again)

⇒ This GRU helps with the vanishing gradient problems.

Date:

Full GRU:

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[\Gamma_r * c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

$$\Gamma_u = \sigma\left(W_u\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$$

$$\Gamma_r = \sigma\left(W_r\left[c^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) * c^{\langle t-1 \rangle}$$

Γ_r = gate relevant tells how relevant the $c^{\langle t-1 \rangle}$ is to compute $\tilde{c}^{\langle t \rangle}$

LSTM:

$$\tilde{c}^{\langle t \rangle} = \tanh\left(W_c\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_c\right)$$

"update" $\Gamma_u = \sigma\left(W_u\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_u\right)$

"forget" $\Gamma_f = \sigma\left(W_f\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_f\right)$

"output" $\Gamma_o = \sigma\left(W_o\left[a^{\langle t-1 \rangle}, x^{\langle t \rangle}\right] + b_o\right)$

$$c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + \Gamma_f * c^{\langle t-1 \rangle}$$

$$a^{\langle t \rangle} = \Gamma_o * \tanh\left(c^{\langle t \rangle}\right)$$

⇒ peep hole connection : variation of LSTM where c_{t-1} is also used to decide the value of all three gate f_t i_t o_t

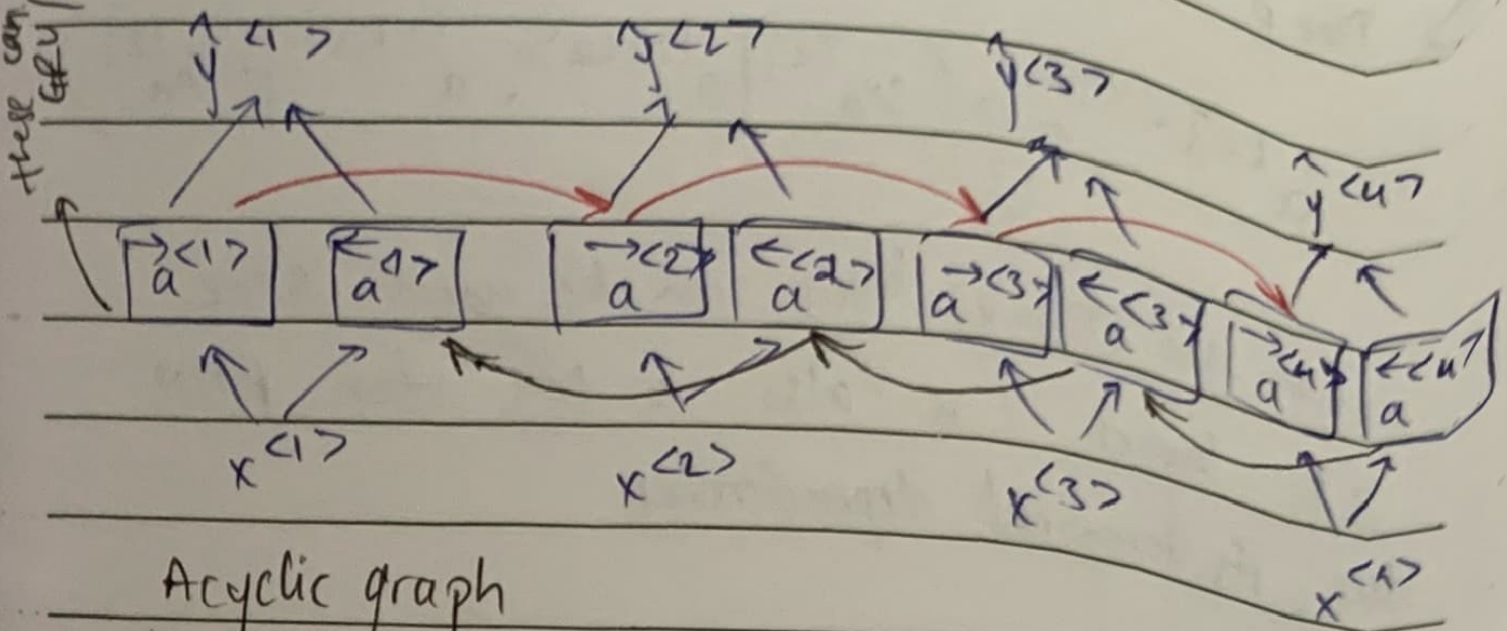
Date:

LSTM in pictures :

Date:

=> Bidirectional RNN (BRNN):

these can be GRU / LSTMs



Acyclic graph

$$\hat{y}^{<1>} = g\left(W_y \left[\overrightarrow{a}^{<t>}, \overleftarrow{a}^{<t>}\right] + b_y\right)$$

This BRNN archi to work, the whole sequence is to be present whole already at the start like NLP applications, whereas won't work for the real world speech applications.

These BRNNs can use tokens from all earlier and later part of the sequence

Date:

⇒ Deep RNNs

$$a^{[2]\langle 3\rangle} = g\left(W_a^{[2]}\left[a^{[2]\langle 2\rangle}, a^{[1]\langle 3\rangle}\right] + b_a^{[2]}\right)$$

Not too many layers can be used since 3/4 layers is also a lot for RNN due to its temporal dependencies.